esa
estec

european space research
and technology centre

Ref: WDI/JG/2105/NL
Issue 4, 20-11-1995

Benchmarking of 32-bit processors for space applications

Prepared by J.Gaisler
Spacecraft Control and Data System Division
Automation and Informatics Department
ESA/ESTEC

Keplerlan 1 - Noordwijk - Netherlands
Mail address: Postbus 299 - 2200 AG - Noordwijk - The Netherlands
Tel: 071-5656565 - Telex: 39098 - Fax: 071-5656040

1 Introduction

In 1992, the European Space Agency begun the development of an 32-bit processing core for embedded space applications. The processing core, ERC32, is based on the SPARC V7 architecture and is developed by a consortium consisting of Matra MHS, Matra Marconi Space and Saab Ericsson Space. At about the same time, Saab Ericsson Space developed a second, proprietary 32-bit processor called THOR. The THOR is a stack based processor also targeted towards embedded applications, and includes hardware support for ADA tasking. Various performance claims have been made but very few actual tests has been performed until now. This evaluation has therefore been performed to measure and verify the performance of the two processors.

The evaluation was performed with the following objectives:

- To measure the computing performance using benchmarks with realistic work-loads, representative of on-board applications.
- To measure the code size of the benchmarks
- To perform the measurements using hardware and software configurations representative of on-board applications

The major conclusions of the evaluation can be summarized in the following points:

- The ERC32 delivers between two and three times higher system performance than THOR.
- The code-size of the benchmarks were the same for ERC32 and THOR.
- The usage of a optimising state-of-the-art compiler is crucial for optimum system performance.

The following paragraphs describe the benchmarked systems and results in detail.

2 T arget systems

2.1 ERC32

The tests have been run on an ERC32 simulator , developed at ESTEC. The IU and FPU has been modelled to execute the instructions in the same number of clock cycles as defined in the ERC32 data sheets [MHS 94]. The execution time of the floating-point instructions is in some cases data-dependent, and in these cases the typical values have been used. The simulator has been characterised against the real ERC32 chip-set and the timing differences are below 0.5% for both integer and floating-point applications. T o provide a reference, the benchmarks were also run on the on a Sparcstation 1. The table below summarises the two systems:

Name	Description	CPU/FPU	CACHE	MEMOR Y
SS1	Sparcstation 1	L64801/W3170 @ 20 MHz	32 KByte	16 MB @ 80 ns
ERC32	ERC32 simulator	90C601E/90C602E @ 14 MHz, 0 ws	0	4MB @ 40 ns

Table 1: Benchmarked SP ARC systems

Two ADA compilers were available for the SP ARC architecture; T elesoft ADA and Gnu ADA (Gnat). The T elesoft compiler is targeted to the Sun4 architecture and could not be used as a cross-compiler for the ERC32. The Gnat however , provides object files which can be linked with libraries for the ERC32 and can therefore be used as a cross-compiler . The following table shows the compile switches:

Compiler	Version	Switches	Comments
GNU ADA	2.07	-O3 -ffast-math	Maximum optimization, run-time checks enabled
Telesoft ADA	4.1	--O prIA	Maximum optimization

Table 2: Compilers and switches - SP ARC

2.2 THOR

The current version of the THOR processor is characterised up to 10 MHz operations. With 40 ns memories and ACT address and data buffers, zero-waitstate operation can be achieved at 8 MHz, operation at 10 MHz will require one waitstate. THOR includes a 256-byte data cache on-chip. The cache is not protected by parity or EDAC checksums and therefore vulnerable for SEU's. The cache may also affect the predictability of the real-time performance. The following THOR configurations have therefore been used:

Description	CPU	CACHE	MEMOR Y
SAAB THOR	THOR @ 8 MHz	256 Byte	0.5MB @40 ns
SAAB THOR	THOR @ 8 MHz, no cache	0	0.5MB @40 ns
SAAB THOR	THOR @ 10 MHz, one waitstate	256 Byte	0.5MB @40 ns
SAAB THOR	THOR @ 10 MHz, one ws, no cache	0	0.5MB @40 ns

Table 3: THOR configurations

An evaluation board for THOR was available but the test were run on the THOR simulator, since some benchmarks did not run on the board. The only available ADA compiler for THOR is Oden ADA, developed by Saab Ericsson Space. There are no optional optimization switches, partial run-time checking is always generated. No C-compiler is available for THOR.

Compiler	Version	Switches	Comments
THOR Oden	0.5.1	-	Run-time checks enabled

Table 4: Oden ADA compiler

3 Benchmarks and results

The benchmarks are divided in two groups, the ESTEC MIX and RUDSTONE. The results are given as execution times for one iteration of the benchmark, even if more iteration have been run in order to obtain a better resolution of the execution time. In a previous version of this document, a benchmark mix known as the 'SAAB MIX' was used. After careful analysis of this benchmark mix, it was found that the benchmark mix has no connection to real applications and gives misleading results. The SAAB MIX will therefore not further be used. As an example, the DAIS benchmark from the SAAB MIX consists of two case-statements, two if-statements, a few assignments and some simple arithmetic operations (add, sub, mul and div). There are no subroutine calls, no array handling, no records or other complex statements which are typical for most applications.

3.1 ESTEC MIX

The ESTEC MIX v1.0 consists of four benchmarks derived from real applications; **kalman, cap, era** and **estec-b**. The kalman and cap benchmarks are taken from the jiawg mix; they consists of kalman filtering (**kalman**) and target capture and tracking (**cap**). The **era** benchmark consists of the joint calculations during free motion movement of ERA, the European Robot Arm. **Estec-b** is a benchmark which performs image compression and decompression using the Estec-B algorithm. All four benchmarks are self-checking, so that any compiler optimization that would affect the result is detected.

System	Kalman	Cap	Era	Estec-B	GEO. MEAN
Sparcstation 1, Telesoft ADA	112	1.8	0.39	48	7.8
Sparcstation 1, GNU ADA	125	2.1	0.36	39	7.8
ERC32 @ 14 MHz, GNU ADA	137	2.0	0.32	44	7.9
THOR @ 8 MHz, cache enabled	410	failed	0.99	failed	failed
THOR @ 8 MHz, cache disabled	761	failed	1.67	failed	failed
THOR @ 10 MHz, 1 ws, cache enabled	457	failed	1.0	failed	failed
THOR @ 10 MHz, 1 ws, cache disabled	1,045	failed	2.2	failed	failed

Table 5: JIA WG benchmark results in milliseconds

The table shows that the performance of ERC32 is approximately equal to a Sparcstation 1. The best THOR system runs about 3 times slower that the ERC32. The difference is too large to depend only on the architectural differences; it is clear that the Oden compiler does not produce very optimized code. It can also be noted that the Telesoft and GNU compilers are identical from a performance point of view. The cost of run-time checking can be substantial, tests showed that pure integer benchmarks such as kalman and estec-b runs about 50% faster without checking. For the floating-point intensive benchmarks, the impact of run-time checking is less noticeable. The **estec-b** and **era** benchmarks failed during compilation and could not be run on THOR.

3.2 Rudstone

The Rudstone benchmark is derived from a satellite ground control system witch performs detection and tracking of orbital objects. The ground control system is developed by Aerospace Corporation (US) and the benchmark was used to compare various ADA compilers. The original benchmark reads sensor data from a file and then performs detection and tracking of satellites. To be able to run the benchmark on embedded targets without file systems, the input data was coded into the program. The original benchmark used double precision floating point calculations; this had to be changed since THOR does not support double precision floats. The change in precision actually affected the detection and tracking results, however as a benchmark the program is still functional. The algorithm of Rudstone uses predominately trigonometric and hyperbolic functions, often using the sqrt() function. The ERC32 includes a hardware sqrt(), and the Rudstone was run both with and without the hardware sqrt() enabled. The hardware sqrt() speeds up the overall program with about 50%. The Rudstone has not yet been run on a THOR system since it is a large application (2.5 Mbyte) and no THOR system with that amount of memory was available.

System	Rudstone (S/W sqrt)	Rudstone (H/W sqrt)
Sparcstation 1, Telesoft ADA	failed	failed
Sparcstation 1, GNU ADA	227	150
ERC32, GNU ADA	156	106
THOR, Oden ADA, 8 MHz, cache enabled	-	-
THOR, Oden ADA, 8 MHz, cache disabled	-	-
THOR, Oden ADA, 10 MHz, 1 ws, cache enabled	-	-
THOR, Oden ADA, 10 MHz, 1 ws, cache disabled	-	-

Table 6: Rudstone execution times in seconds

3.3 Code size measurement

Special precautions were made when the code size was measured. To remove any size differences in the TEXT_IO library, a version without TEXT_IO was derived of each benchmark. Since different compilers provide different amount of run-time support, a “null” program was first compiled and the size was then subtracted from the code size of the benchmarks. With this approach, only the size of the benchmark code is measured, excluding all run-time support. This is especially important for small benchmarks, such as the ESTEC-MIX, which are often smaller than the run-time library. In a real application, the overhead of the run-time library is less noticeable.

The SP ARC V7 architecture does not include a multiplication or division instruction. Instead, these routines are performed in software. The code size overhead of the multiplication and division routines can be significant in small programs, such as the ESTEC-MIX, but is negligible in programs with realistic size. When the code size of the ESTEC-MIX was measured, the individual benchmarks were therefore combined in one program.

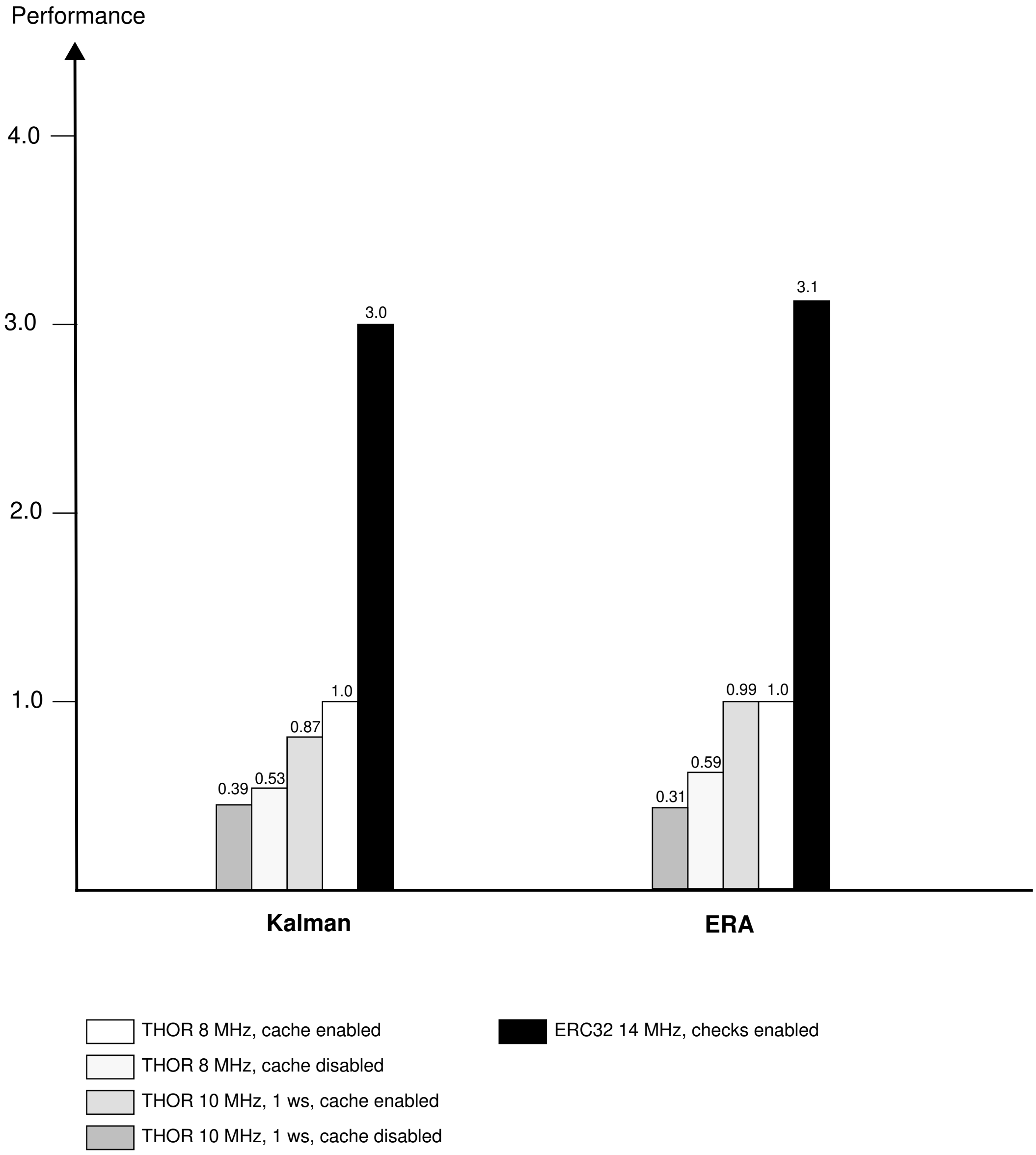
Program	ESTEC-MIX	Rudstone	Total
SPARC, Telesoft ADA	73,728 + 22,576	failed	failed
SPARC, Gnu ADA	53,632 + 23,984	116,360 + 436,432	630,408
THOR, Oden ADA	failed	137,228 + 437,436	failed

Table 7: Code size measurements (code + data) in bytes

Since the ESTEC-MIX did not compile completely on THOR, only the Rudstone could be used for code size comparison. For the Rudstone benchmark, the code size generated by GNU ADA and Oden ADA are approximately equal. The Telesoft compiler failed compiling Rudstone. Note that all used benchmarks were purely sequential, i.e. no tasking was used. When tasking is used, a much larger part of the run-time system has to be included, and the total code size depends strongly on the implementation of the kernel. Also, the required stack size was not included in the code size since it was not possible to obtain this figure from GNU ADA.

4 Conclusions and comments

The following table summarizes the measured performance for kalman and era, no other results were available for THOR. The results have been normalised relative to the best THOR value.



It is difficult to make a proper analysis of the performance of the two processors, since only a few benchmarks could be executed on THOR. Nevertheless, the performed test indicate a clear advantage for ERC32, which delivers up to 3 times higher system performance than THOR. The code size measurements indicate that the Oden and GNU ADA compilers generated approximately the same code size for a given program, run-time system excluded. It can be assumed that the run-time system for THOR is smaller than for ERC32, since parts of the tasking mechanisms are in hardware. Whether this difference is significant compared to the size of a typical application is yet to be studied.

During this benchmarking exercise, it has become apparent that computational requirements can never be expressed in MIPS figure using a particular assembly level instruction mix. For the ERC32, benchmarks were compiled and run both with and without optimization. The assembly level instruction mix was equal in both cases, even though the un-optimized benchmarks executed more that twice as slow . Performance requirement for systems where high-level languages are to be used should therefore be expressed as a maximum execution time of a test program written in the same language. It is also important that the selected test program uses similar language elements and calculations as the target application.

References:

[Gomez 91] M. de Jong and F . Gomez-Molinero. "Benchmarking of compilers for space embedded real-time systems", ESA-STR-223, ESA/ESTEC 1991

[Saab 90] Saab Space AB. "RISC Evaluation study - final report" RIS/TRP/0010/ SAAB, 1990

[MHS 90] Matra MHS SA,."IU-RT device specification", "FPU-RT device specification", Issue 7, 1995